```
#include <alloc.h>

struct reqblock
{ int block;
   struct reqblock *next;
} *first,*curr,*prev;

int n,currpos,headmove=0;
char direction;

void get_req_blocks();
 void scan();
void main()
{
  clrscr();
   printf("\nEnter total number of blocks in the disk:");
   scanf("%d",&n);
   printf("\nEnter request block numbers string terminated by -1\n");
    get_req_blocks();
    //print req string
    curr=first;
    while(curr!=NULL)
    { printf("%d\t",curr->block);
      curr=curr->next;
     }
     printf("\nEnter direction of head movement:(F-Forwad,B-Backward):");
     flushall(); scanf("%c",&direction);
      printf("\nEnter block no. as current head position:");
      scanf("%d",&currpos);
     scan();
     printf("\nNumber of headmovements:%d",headmove);
  }

  void get_req_blocks()
  { struct reqblock *t,*pt;
    int blockno;
    first=NULL;
    scanf("%d",&blockno);
    while(blockno!=-1)
    { curr=(struct reqblock *) malloc(sizeof(struct reqblock));
      curr->block=blockno;
      curr->next=NULL;
     if (first==NULL) //req str is empty
         first=curr;
     else
         prev->next=curr;
      prev=curr;
      scanf("%d",&blockno);
    }//while
  }

 void scan()
  {  int selblock;
     printf("\nList of request served:\n");

     while(first!=NULL)
     {
       if (!look())
         direction=(direction=='F')?'B':'F';
       selblock=get_next_block();
       if (selblock!=-1)
           printf("%d\t",selblock);
        if (direction=='F')
        {  if (currpos==n-1)
             direction='B';
         else
         {  currpos++; headmove++;
           }
      }
      else
      {
```

```c
         if (currpos==0)
             direction='F';
         else
         { currpos--;  headmove++;}
       }
      }//while
      headmove--;
  }

int look()
{
  if (direction=='F')
  {  curr=first;
     while(curr!=NULL)
     {   if (curr->block>currpos)
         return(1);
         curr=curr->next;
      }
      return(0);
    }
  else //direction='B'
  {   curr=first;
     while(curr!=NULL)
     {  if (curr->block<currpos)
         return(1);
         curr=curr->next;
      }
      return(0);
    }
}

get_next_block()
  {
        int selblock;
         curr=first;
          while(curr->block!=currpos)
           {  prev=curr;
              curr=curr->next;
              if (curr==NULL) return(-1);
            }
            selblock=curr->block;
            if (curr==first)
               first=first->next;
             else
               prev->next=curr->next;
             free(curr);
            return(selblock);
 }
```